THOR: Secure Transformer Inference with HE

ACM CCS 2025

Jungho Moon, Hanyang University & Desilo Inc.

Dongwoo Yoo, Yonsei University

Xiaoqian Jiang, UTHealth at Houston

Miran Kim, Hanyang University

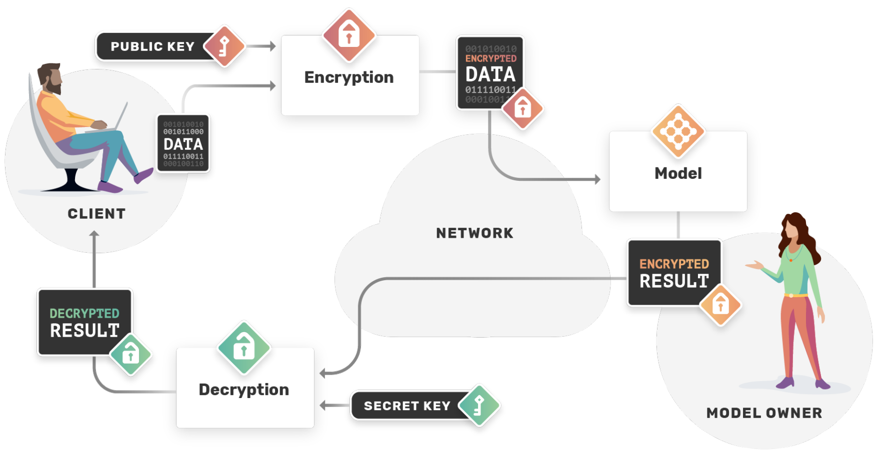
Motivation

- Cloud AI services require users' personal data for inference and analysis
  - **Data abuse**: if data is available, it will be used

Motivation

- Goal: Trustworthy ***Machine Learning as a Service***

  (Privacy-preserving Personalized Prediction Services)



Homomorphic Encryption and LLM : Is ChatGPT end to end encrypted ?

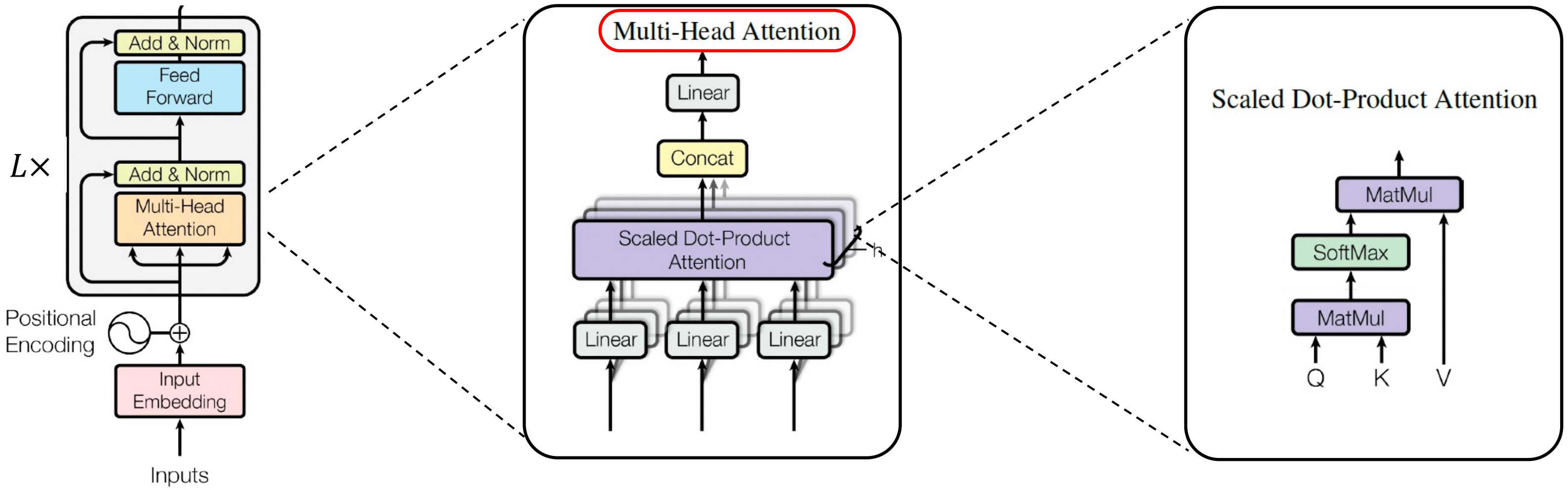
April 19, 2023 — Rand Hindi

This is the first post in a series dedicated to making large language models (LLMs) encrypted end-to-end with homomorphic encryption. We will publish more details on how to achieve it technically as we make progress towards this goal in the coming years.

Since most of the challenges in FHE are already solved (or will be in the near future), **we can confidently expect to have end-to-end encrypted AI within 5 years.** I strongly believe that when this happens, nobody will care about privacy anymore, not because it's unimportant, but because it will be guaranteed by design.

Transformer-based Model

- What is *Transformer*?
  - Self-attention based architecture [V+18]
  - Foundation of modern NLP models like BERT, GPT, T5, BART
  - Parallelizable and efficient processing of sequences ⇒ ***matrix computation***



[V+18] Attention is all you need, NeurIPS'18

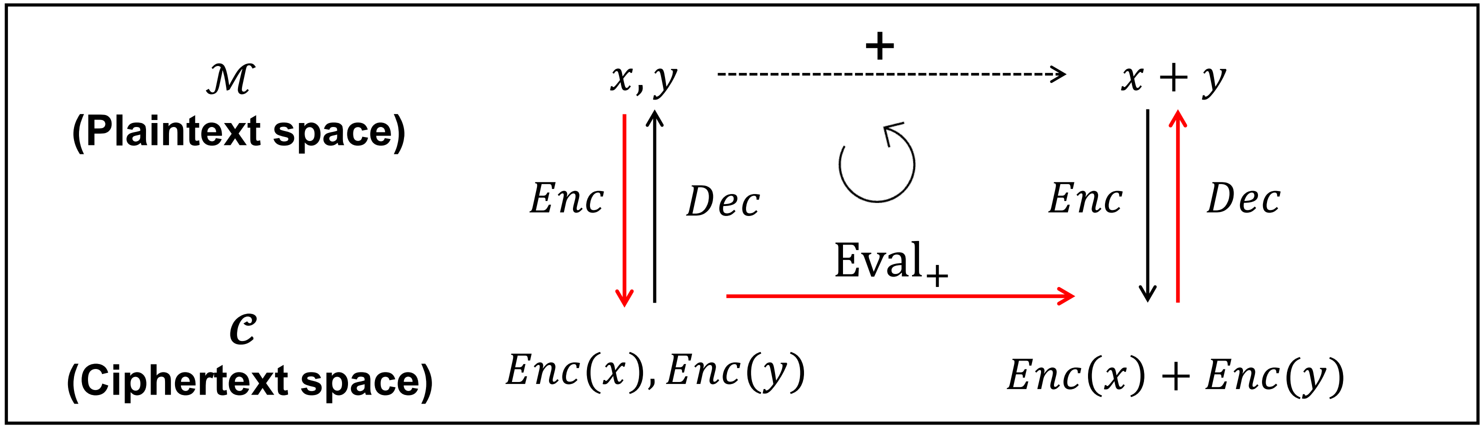
What is Homomorphic Encryption (HE)?

- Traditional encryption protects only storage and transmission—not the computation phase
- HE can evaluate arithmetic functions on encrypted data.
  - For $x, y \in \mathcal{M} = \mathbb{Z}_q[X]/(X^N + 1)$,

$$Dec\big(Enc(x) + Enc(y)\big) = Dec\big(Enc(x)\big) + Dec\big(Enc(y)\big) = x + y$$

$$Dec\big(Enc(x) * Enc(y)\big) = Dec\big(Enc(x)\big) * Dec\big(Enc(y)\big) = x * y$$

  - SIMD-style *vectorized* operations (: Entry-wise adds & mults, rotations between slots)



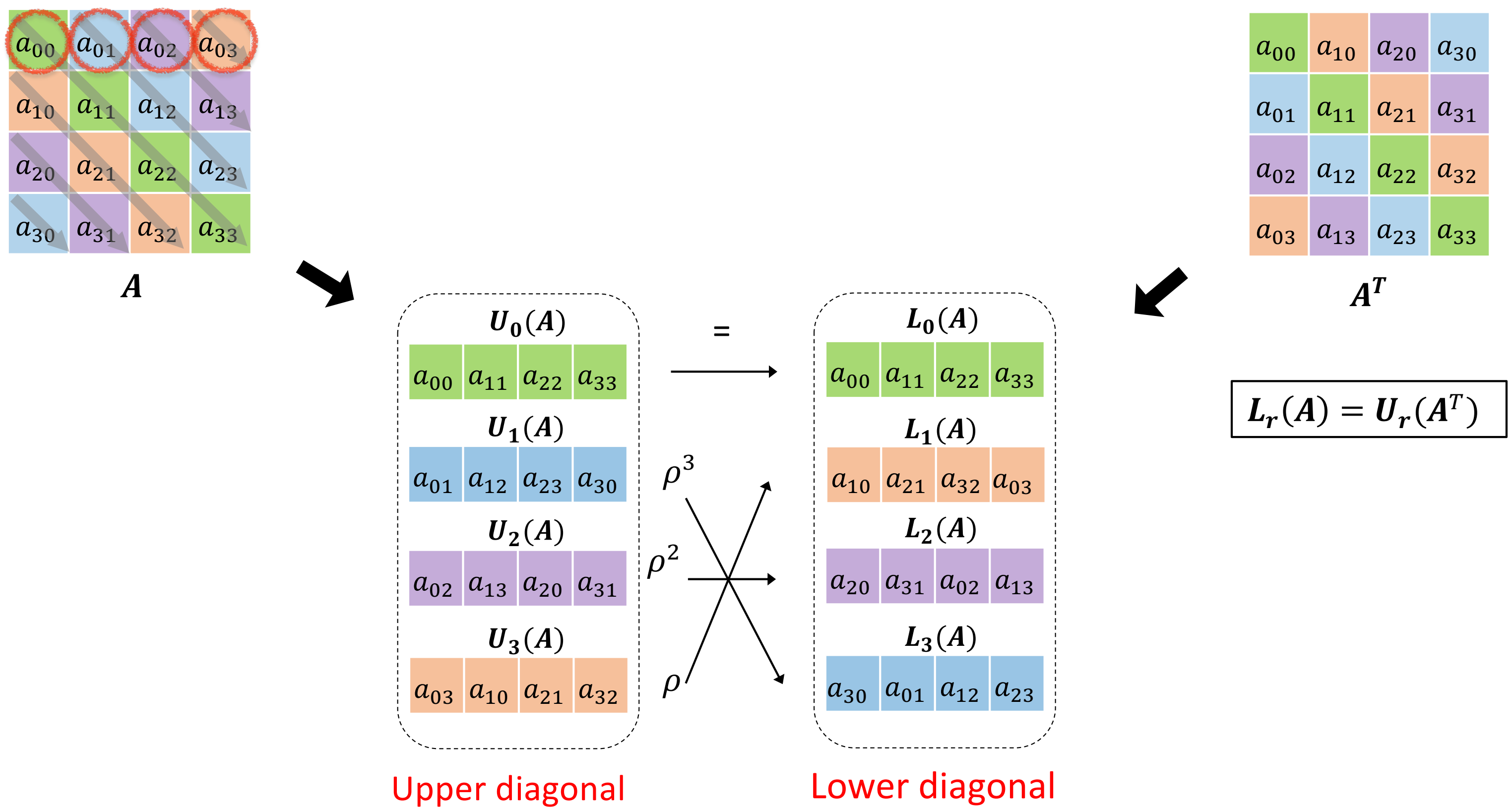
Homomorphic Matrix Computation

Question. How to efficiently perform matrix computation over encrypted data?
That is, how to encode matrix into $1d$-vector and
express matrix multiplication as HE operations?

- Related Work
  - *Diagonal*-major encoding: HS14 (Mat×Vec)
  - *Column*-major encoding: BOLT (PC-MM)
  - *Row*-major encoding: JKLS'18
    - Small matrix size assumption: $d^2 \leq s$ (s: vector size)
    - Inefficient for PC-MM (e.g., linear projection, feed-forward)
    - Matrix transposition: $\mathcal{O}(d)$ complexity

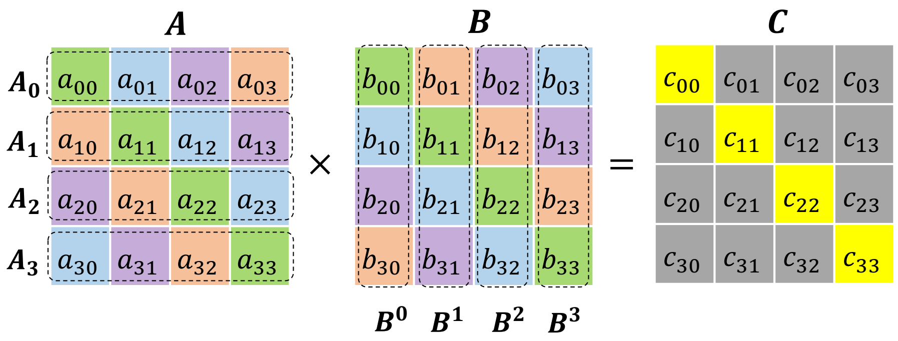
[HS14] S. Halevi, V. Shoup. "Algorithms in HElib", CRYPTO 2014
[JKLS18] X. Jiang, **M. Kim**, K. Lauter, Y. Song. "Secure outsourced matrix computation and application to neural networks", CCS'18

Diagonal-major Matrix Encoding

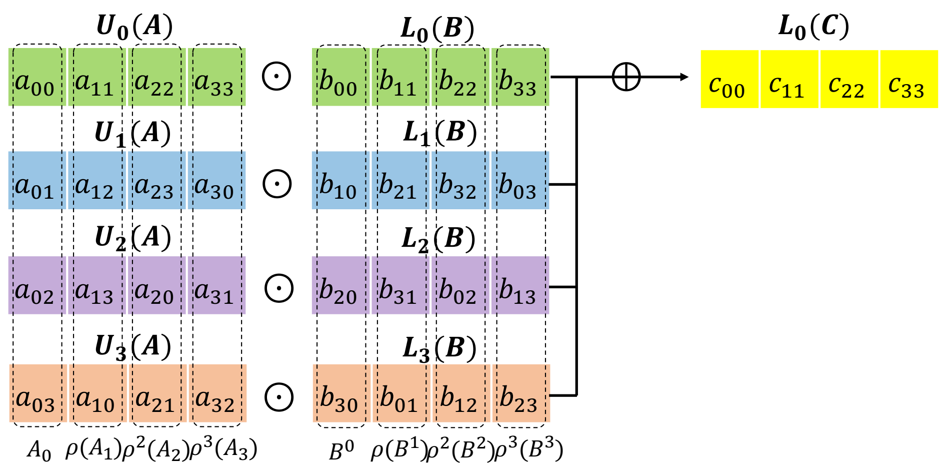


$$\boxed{L_r(A) = U_r(A^T)}$$

Main Idea: A New Matrix Multiplication



- $L_0(C)$: the 0th lower diagonal *ρ: left-rotation
  - $c_{00} = \langle A_0, B^0 \rangle$
  - $c_{11} = \langle A_1, B^1 \rangle = \langle \rho(A_1), \rho(B^1) \rangle$
  - $c_{22} = \langle A_2, B^2 \rangle = \langle \rho^2(A_2), \rho^2(B^2) \rangle$
  - $c_{33} = \langle A_3, B^3 \rangle = \langle \rho^3(A_3), \rho^3(B^3) \rangle$



$$\boxed{\begin{aligned} L_r(C) &= \left(C_{r,0}, C_{r+1,1}, \ldots, C_{r-1,d-1}\right) \\ &= \sum_{l=0}^{d-1} \rho^r\left(U_{l-r}(A)\right) \odot L_l(B) \end{aligned}}$$

HE-friendly Matrix Multiplication

$$\boxed{\begin{aligned} L_r(C) &= \left(C_{r,0}, C_{r+1,1}, \ldots, C_{r-1,d-1}\right) \\ &= \sum_{l=0}^{d-1} \rho^r\left(U_{l-r}(A)\right) \odot L_l(B) \end{aligned}}$$

(PC-MM) $[\![L_r(\boldsymbol{C})]\!] = \sum_{l=0}^{d-1} \mathrm{Mult}(\rho^r(U_{l-r}(\boldsymbol{A})), [\![L_l(\boldsymbol{B})]\!])$

(CC-MM) $[\![L_r(\boldsymbol{C})]\!] = \sum_{l=0}^{d-1} \mathrm{Mult}(\rho^r([\![U_{l-r}(\boldsymbol{A})]\!]), [\![L_l(\boldsymbol{B})]\!])$

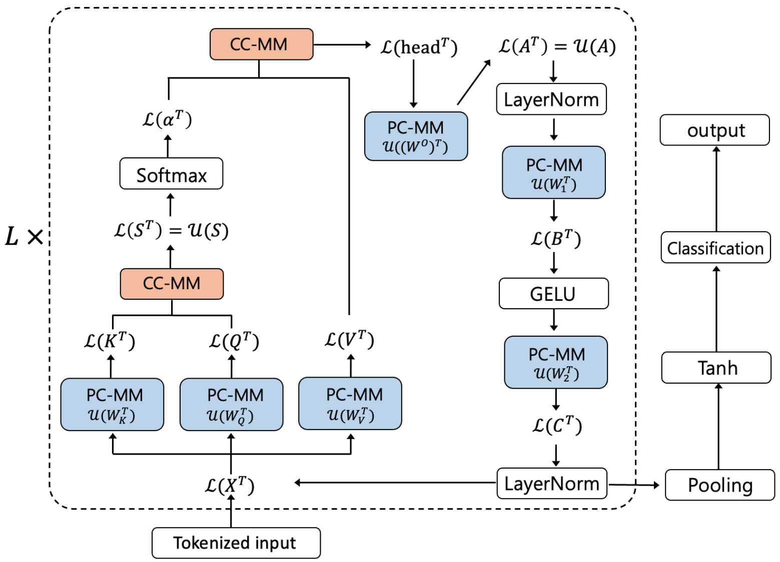
- ✓ Easy to implement
- ✓ Optimized for both PC-MM and CC-MM
- ✓ Matrix transposition for free but a different format
- ✓ *Unified* encrypted matrix representation (reusable for subsequent computations)
- ✓ Easily extended to parallel matrix multiplication (interlace multiple matrices & batch the computation)
- ✓ Scalability to *parallel* large-scale MM

CC-MM Computation of $A^{(z)}B^{(z)}$ for $1 \le z \le H$, where $A^{(z)}, B^{(z)} \in \mathbb{R}^{128\times128}$ and $s = 2^{14}$

	Equation	H				
		1	2	4	8	16
JKLS'18	$H(5\sqrt{n} + 4n)$	572	1144	2272	4576	9152
Our work	$H(n/2+3) + n/4(\log(n/H)+3)$	387	422	520	760	1264

THOR: Secure Transformer-based Inference

- BERT-base model (L =12 layers, 110M parameters, n =128 tokens)
- **10 minutes** with only a **0.8% accuracy drop**
  - **16x** improvement over NEXUS (2.7hr)



[Z+25] "Secure transformer inference made noninteractive", NDSS'25

Intel Xeon Platinum 8462 at 2.8GHz, A100 GPU (DESILO FHE library)

Operation	Input	Time (sec)
Attention layer	$3 \times (\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times64})$	49.77
Attention score	$12 \times (\mathbb{R}^{128\times64} \times \mathbb{R}^{64\times128})$	16.25
Softmax	$12 \times (\mathbb{R}^{128\times128})$	15.53
Attention head	$12 \times (\mathbb{R}^{128\times128} \times \mathbb{R}^{128\times64})$	13.08
Multi-head attention	$\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times768}$	27.43
LayerNorm1	$\mathbb{R}^{128\times768}$	7.13
FC1	$\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times3072}$	49.80
GELU	$\mathbb{R}^{128\times3072}$	29.42
FC2	$\mathbb{R}^{128\times3072} \times \mathbb{R}^{3072\times768}$	49.19
LayerNorm2	$\mathbb{R}^{128\times768}$	4.10
Pooler & Classification	$\mathbb{R}^{128\times768}$	2.70
Bootstrappings	-	337.86
Total	-	**602.26**

Dataset	#Test	Metric	Unencrypted				Encrypted
			Baseline	G	G-LN	G-LN-S	
MPRC	408	Accuracy	85.29	85.54	85.54	85.78	**84.80**
		F1-score	89.90	90.05	90.05	90.24	**89.49**
RTE	277	Accuracy	72.20	71.48	71.84	72.20	**71.12**
SST-2	872	Accuracy	91.51	91.40	91.40	91.63	**90.71**

Summary

- New Efficient *parallel* matrix computation
  - Optimizations: Block-wise PC-MM / BSGS-integrated CC-MM
  - Row-wise computation over diagonals
- Nonlinear approximation
  - Softmax: square-and-normalization approach
  - LayerNorm, GeLU, Tanh: use the Goldschmidt's algorithm + Adaptive Successive Over-Relaxation method (aSOR)
- *End-to-end* secure inference

Implementation available at:
`https://github.com/crypto-starlab/THOR`

ia.cr/2024/1881